View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Electronic Archive of Kyiv Polytechnic Institute

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Система збору та обробки електронних петицій на основі Ruby on Rails»

Виконав:
студент IV курсу, групи ІТ-51 Олійник Олександр Сергійович _____

Керівник:
Професор кафедри АУТС, д.т.н., доцент Корнієнко Б. Я. _____

Рецензент:
Посада, науковий ступінь, вчене звання Прізвище, ініціали _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2019 рік

Пояснювальна записка
до дипломного проекту
на тему: «Система збору та обробки електронних петицій на основі Ruby on Rails»

Київ – 2019 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

ЗАВДАННЯ
на дипломний проект студенту
Олійнику Олександру Сергійовичу

1.Тема проекту «Система збору та обробки електронних петицій на основі Ruby on Rails», керівник проекту Корнієнко Богдан Ярославович, професор кафедри АУТС, д.т.н., доцент, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту 18.05.2019

3. Вихідні дані до проекту
_____
_____

4. Зміст пояснювальної записки
_____
_____

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)
_____
_____-
_____

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка

Студент О. С. Олійник

Керівник проекту Б. Я. Корнієнко

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
	4	КПІ.ІП-5201. ХХХХХХ.02.81	Комплекс задач розгортання програмних продуктів з використанням платформи Google Cloud. Технічне завдання		
	4	КПІ.ІП-5201. ХХХХХХ.02.81	Комплекс задач розгортання програмних продуктів з використанням платформи Google Cloud.Пояснювальна записка		
	4	КПІ.ІП-5201. ХХХХХХ.03.81	Програма та методика тестування.		
	4	КПІ.ІП-5201. ХХХХХХ.04.51	Керівництво користувача.		
	4	КПІ.ІП-5201. ХХХХХХ.04.51	Керівництво адміністратора.		
	3	КПІ.ІП-5201. ХХХХХХ.04.51	Діаграма сценаріїв використання.		
	3	КПІ.ІП-5201. ХХХХХХ.04.51	Діаграма структури бази даних		
	3	КПІ.ІП-5201. ХХХХХХ.04.51	Діаграма бізнес-процесу потоку інтеграції та доставки		

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з трьох розділів, містить 3 таблиці, 3 додатків та 15 джерел – загалом 63 сторінок.

Об`єкт дослідження: веб системи, що використовуються для створення та обробки електронних петицій на базі Ruby on Rails.

Мета дипломного проекту: створити легко розширюваний веб застосунок, здатний до швидкого та зручного створення петицій користувачем, голосування зі петицій інших користувачів та відправлення петиції до установи.

У першому розділі було розроблено архітектуру веб застосунку. Побудовано структурну схему класів та діаграму послідовності.

У другому розділі проведено тестування веб застосунку за розробленим планом тестування. Описано процес тестування.

У третьому розділі описано розгортання та впровадження веб застосунку, а також наведено схему структурну розгортання.

У додатках наведено: опис програми, схема структурна класів програмного забезпечення, схема структурна послідовності виконання.

КЛЮЧОВІ СЛОВА: ПЕТИЦІЇ, СИСТЕМА, ВЕБ-ЗАСТОСУНОК

ABSTRACT

Explanatory note of the diploma project consists of 4 sections, 3 annexes, 3 tables and 15 sources – total 63 pages.

The object of study: web systems used to create and process electronic petitions based on Ruby on Rails.

The aim of the diploma project: to create an easily expandable web application, capable of fast and convenient creation of petitions by the user, voting from petitions of other users and sending petitions to the institution.

In the first section, the web application architecture was developed. A structured scheme of classes and sequence diagrams was constructed.

The second section tests the web application for the developed test plan. The testing process is described.

The third section describes the deployment and implementation of the Web application, as well as the structured deployment scheme.

The appendixes contain: description of the program, the diagram of the structural classes of the software, a diagram of the structure of the execution.

KEYWORDS: PETITIONS, SYSTEM, WEB-APPLICATION

ЗМІСТ

ДОДАТОК В СХЕМА БАЗИ ДАНИХОШИБКА! ЗАКЛАДКА НЕ ОПРЕДЕЛЕНА.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Ruby – інтерпретована мова програмування високого рівня загального призначення.

Rails – являє собою платформу веб-додатків на сервері, написану на Ruby.

Model-View-Controller (MVC) – це архітектурна схема, яка зазвичай використовується для розробки інтерфейсів користувача, що розділяє додаток на три взаємопов'язані частини..

MySQL – є відкритою системою управління реляційними базами даних.

Capistrano – це інструмент з відкритим кодом для запуску сценаріїв на декількох серверах; його основне використання - розгортання веб-додатків.

Bootstrap – це безкоштовна та відкрита CSS-фреймворк, орієнтована на адаптивну веб-розробку для мобільних пристроїв.

Javascript – мова програмування веб-застосунків.

Jquery – бібліотека створена на основі мови програмування Javascript.

HTML – є стандартною мовою розмітки для документів, призначених для відображення у веб-переглядачі.

CSS – це таблиця стилів, яка використовується для опису презентації документа, написаного на мові розмітки, наприклад HTML [13].

BankID – спосіб верифікації громадян через українські банки для надання адміністративних чи інших послуг [12].

API – це набір визначень підпрограм, протоколів зв'язку та інструментів для побудови програмного забезпечення.

ВСТУП

З кожним днем все більше сервісів переходить з автономної роботи в онлайн, наприклад магазини, банки, державні сервіси, це дає змогу швидше та зручніше отримувати послуги користувачам.

У будь-якій країні світу люди прагнуть жити краще. Але влада не завжди бачить всі проблеми, які турбують населення та знижують якість життя. Реакція населення, яке було не задоволене владою та їх роботою спочатку існувало як громадський рух, та лише згодом це отримало конституційно-правове закріплення з урахуванням особливостей тих чи інших держав, саме так виникла петиція. Отже, петиція – це одна з форм звернення громадян до влади з проханням, яке було викладене в петиції. Сьогодні багато держав у світі мають впроваджену систему петицій. Для того, щоб ваша петиція потрапила на розгляд до влади, необхідно, щоб вона отримала необхідну кількість підписів громадян цієї країни. Донедавна люди писали петиції на аркушах паперу, збирали підписи та відправляли до органів влади через пошту, але зараз все це відбувається в мережі інтернет. Петиції також можливо направляти до підприємств або великих компаній, які займаються виробництвом товарів, техніки або надають послуги. Наприклад, якщо компанія, яка випустила техніку при виробництві техніки не врахувала всі чинники, які буду докучати користувачу використовувати з комфортом пристрій або програмне забезпечення, яке на ньому встановлено, то користувач може написати петицію та вказати фактори, які йому докучають. Якщо, ця петиція отримає достатню кількість підписів від інших користувачів, які також мають ці проблеми або згодні зі змістом петиції, тоді ця петиція потрапить на розгляд до компанії і вона зможе розглянути цю петицію, та відповісти користувачам. Для компанії це буду корисно зрозуміти де компанія зробила помилку, та не врахувала всі можливі помилки, та зможе його виправити відразу, якщо є така можливість або в наступній партії продукту.

Завданням дипломного проекту є створення зрозумілої та зручної для користувача системи збору та обробки електронних петицій. Головною відмінністю системи від аналогів є можливість приймати від користувачів петицію, яку вони хочуть направити не тільки до будь-якого органу влади, але й до організації з проханням або вимогою. Система повинна повністю автоматизовано приймати та обробляти петиції, а саме: збирати підписи, відстежувати до якого часу петиція активна, архівувати або відправляти до вказаного адресату, мати два типи користувачів, що подає петиції та відповідає. Користувач повинен зареєструватися та бути авторизованим у системі, щоб мати можливість створювати петиції та підтримувати петиції інших користувачів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Веб застосунок спрямований на створення єдиної відкритої системи створення та обробки електронних петицій. Система повинна давати можливість користувачу зареєструватися та створити петиції, яку інші користувачі мають можливість підтримати; кабінет користувача, де користувач зможе відслідковувати петиції, які він створив та підтримав; кабінет компаній або державних структур, де користувачі зможуть побачити петиції, які до них спрямовані та дати на них відповідь, яку потім всі зможуть переглянути. Також система повинна мати архів петицій, які були створені користувачами, петиції повинні мати статуси, щоб користувач зміг швидко зрозуміти на якому етапі знаходиться петиція.

Для забезпечення швидкої та комфортної взаємодії, програмний продукт буде виконано з використанням фреймворку Rails, який написаний на мові програмування Ruby, з використанням концепції MVC. Суть концепції MVC розділити данні додатку, інтерфейсу користувача та управлінням логіки на три незалежні компоненти. Зберігати дані користувачів та петиції в базі даних MySQL. Розгорнути веб застосунок на сервері, щоб додаток міг витримувати великі навантаження від користувачів та буди завжди в робочому стані.

1.2 Змістовний опис і аналіз предметної області

Системи для збору та обробки онлайн петицій повинні дати користувачу можливість швидко, зручно та без зусиль створити петицію, з вимогою або побажанням, та відправляти до отримувача, якого визначить користувач. Також вона повинна підтримувати можливість реєстрації користувача з використанням API єдиної системи BankID, що дає можливість верифікувати користувачів з

України за допомогою даних, які вже є банківських базах даних. Так як користувачів, які будуть відвідувати веб застосунок буде багато, система повинна бути оптимізована під виконання великої кількості задач користувачів та витримувати навантаження.

1.3 Аналіз IT-проектів за схожим принципом роботи

Серед систем та сервісів для створення та обробки електронних петицій маємо наступних конкурентів.

1. Електроні петиції офіційне інтернет-представництво Президента України – https://petition.president.gov.ua.

Офіційний сайт президента України, на якому кожен бажаючий може розмістити петицію до Президента України. Почнемо з самого початку, якщо користувач хоче створити петицію, йому необхідно зареєструватися, реєстрація на сервісі реалізована дуже зручно та відбувається швидко. Зареєструватися можливо за допомогою електронної пошти, далі необхідно підтвердити свою особу, на цьому сервісі це робиться дуже зручно за допомогою паспортних даних або BankID. Саме технологія BankID робить реєстрацію на сервісі дуже зручною, тому що користувачу не потрібно шукати свої документи, а потім вводити їх в поля реєстрації на сайті, необхідно лише через банк підтвердити свою особу і всі дані користувача будуть автоматично передані до сервісу та він зможе лиши або підтримати петиції. Додання петиції також дуже швидке, необхідно заповнити лише два поля для вводу тексту.

2. Електронні петиції – Верховна Рада України – https://itd.rada.gov.ua/services/petitions.

Офіційний сайт Верховної Ради України, де також кожен громадянин України може залишити електрону петицію. Але цей сервіс не має переваг або гарних рішень, які здатні спростити взаємодію користувача зі системою. Сервіс

не має фільтрів для зрозумілого та швидкого сортування петицій, реєстрація вимагає заповнення всіх текстових полів інформацією з документів користувача. Система є застарілою та не зручною.

3. Єдина система місцевих петицій – https://www.e-dem.in.ua.

Сервіс для відправки петицій до місцевої влади всіх областей України. Користувач має можливість відправити до місцевої влади своєї області або інших, з пропозицією або проханням, щоб змінити то, що його не задовольняє. Дуже зручний сервіс, так як користувач може з одного місця (сайту) відправити петицію до місцевої ради будь-якої області України. Сервіс також має зручну реєстрацію та авторизацію за допомогою систем BankID, GovID. Має фільтри, які допоможуть користувачу відшукати потрібні йому петиції, також система маю приємний та легкий дизайн, яким користувачу буде приємно та комфортно створювати або редагувати петиції, які він буде створювати.

4. Платформа для створення петицій в Європі openPetition – https://www.openpetition.eu.

OpenPetition підтримує людей які створюють петиції у підготовці їх клопотання, зборі підписів і подачі клопотання відповідного одержувача. Крім того, openPetition вимагає думки парламентарів незалежно від формального процесу подання петиції. Це стосується збереження молодіжного клубу, виведення з експлуатації атомних електростанцій, змін у законодавстві або будівництва вітрових турбін: щодня користувачі починають петиції на нашій платформі та просувають зміни: місцеві, регіональні, загальнонаціональні та покроково в Європі. Понад 6 мільйонів людей користуються відкритою петицією та стають частиною цифрової демократії. Система дає змогу при створенні петиції користувачу детально вказати в якому регіоні виникла проблема, суть проблеми, як потрібно вирішити проблему, контакти користувача, який подає петицію та можливість завантажувати файли.

Висновок до розділу

У цьому розділі було описано та проаналізовано предметну область розробки. Було виділено подібні системи у обраній області та порівняно між собою. Для розробки системи було обрано найкращі частини готових систем, з метою створити покращену систему, яка не буде мати недоліків всіх існуючих рішень і також містити в собі їх переваги:

- зручну та швидку для користувача реєстрацію;
- можливість створення петицій та відправка їх не лише до органів влади, але й до підприємств або компаній;
- підтримання петицій іншими користувачами (голосування);
- зручна система фільтрів, для зручного та гнучкого сортування петицій;
- архів петицій;
- відображення поточного статусу петиції.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Розробка система виконувалась на основі Ruby on Rails з використанням HTML, CSS, JavaScript та MySQL. Серверна часина розроблена на мові програмування Ruby з фреймворком Rails, базою для зберігання даних було використано MySQL. При розробці системи був взятий паттерн програмування MVC (Рисунок 2.1.1). MVC є шаблоном для архітектури прикладного програмного забезпечення.

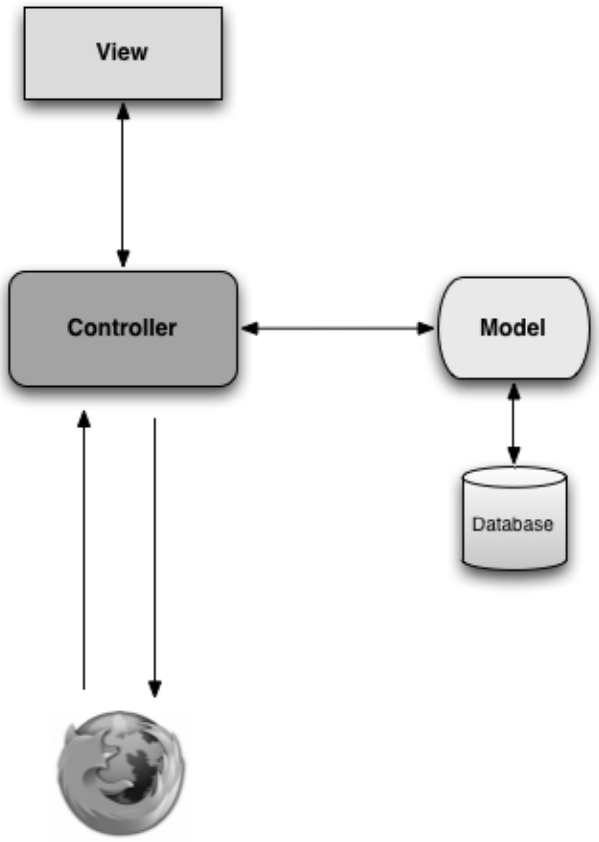

Рисунок 2.1.1 – Схематичне представлення архітектури моделі-перегляду-контролера (MVC)[2]

Він розділяє додаток на наступні компоненти:

– моделі для обробки даних і бізнес-логіки;

– контролери для обробки інтерфейсу користувача та програми;

– відображення для обробки графічних об'єктів інтерфейсу користувача та презентації.

Компонент Model відповідає всім логічним даним, з яким користувач працює. Це може представляти або дані, які передаються між компонентами View і Controller, або будь-які інші дані, пов'язані з бізнес-логікою. Наприклад, об'єкт клієнта буде отримувати інформацію про петицію з бази даних, маніпулювати ним і оновлювати дані назад до бази даних або використовувати його для передачі даних.

Контролери діють як інтерфейс між компонентами Model і View для обробки всієї бізнес-логіки і вхідних запитів, маніпулювання даними за допомогою компонента Model і взаємодії з Views для відображення кінцевого виводу. Наприклад, контролер петицій буде обробляти всі взаємодії і входи від відображення користувача і оновлювати базу даних, використовуючи модель петиції. Цей же контролер буде використовуватися для перегляду даних петиції.

Як і Perl, Ruby добре обробляє текст. Як і Smalltalk, все в Ruby є об'єктом, і Ruby має блоки, ітератори, мета-класи та інші хороші речі. Ви можете використовувати Ruby для написання серверів, експериментувати з прототипами і для щоденних завдань програмування. Як повністю інтегрований об'єктно-орієнтована мова, Ruby добре масштабує. Кожне значення в Ruby є об'єктом, навіть найпримітивнішим: рядки, числа і навіть істинні і помилкові. Кожен об'єкт має клас, і кожен клас має один суперклас. У корені класової ієрархії знаходиться клас BasicObject, від якого всі інші класи, включаючи Object, успадковуються. Кожен клас має набір методів, які можна викликати на об'єктах цього класу. Методи завжди називаються в об'єкті – не існує "методів класу", як це робиться на багатьох інших мовах (хоча Ruby робить велику роботу з підроблення). Кожен

об'єкт має набір змінних екземплярів, які містять стан об'єкта. Екземплярні змінні створюються і доступні з методів, які викликаються об'єктом. Змінні екземпляра повністю приватні для об'єкта. Жоден інший об'єкт не може бачити їх, навіть інші об'єкти одного класу або сам клас. Весь зв'язок між об'єктами Ruby відбувається за допомогою методів.

Ruby on Rails (або просто «Rails» для короткого викладу) – це структура веб-розробки, написана на мові програмування Ruby. З моменту свого дебюту в 2004 році Ruby on Rails швидко став одним з найпотужніших і популярних інструментів для створення динамічних веб-додатків. Rails використовується компаніями, як Airbnb, Basecamp, Disney, GitHub, Hulu, Kickstarter, Shopify, Twitter і Yellow Pages. Є також багато магазинів веб-розробки, які спеціалізуються на Rails, таких як ENTP, thoughtbot, Pivotal Labs, Hashrocket і HappyFunCorp, а також безліч незалежних консультантів, тренерів і підрядників.

Що робить Rails настільки великим? Перш за все, Ruby on Rails є 100% відкритим вихідним кодом, доступним під щедрою ліцензією MIT, і в результаті він також не коштує нічого для завантаження або використання. Rails також багато в чому завдячує своєму елегантному і компактному дизайну; використовуючи гнучкість базової мови Ruby, Rails ефективно створює доменну мову для написання веб-додатків. Як наслідок, багато загальних завдань веб-програмування, таких як генерування HTML, створення моделей даних і URL-адрес маршрутизації, легко виконуються за допомогою Rails, а результуючий код програми є коротким і зрозумілим.

Rails також швидко адаптується до нових розробок у сфері веб-технологій та дизайну каркасів. Наприклад, Rails був одним з перших фреймворків для повної переробки та реалізації архітектурного стилю REST для структурування веб-додатків (про які ми дізнаємося протягом всього навчального посібника). І коли інші рамки розробляють успішні нові технології, творець Rails Девід Хайнемейер Хансон і основна команда Rails не соромляться включати свої ідеї. Можливо, найбільш драматичним прикладом є злиття Rails і Merb, конкуруючих

веб-фреймворків Ruby, так що Rails тепер отримує переваги від модульної конструкції Merb, стабільного API і покращеної продуктивності.

Нарешті, Rails отримує вигоду від надзвичайно захопленої та підтримуючої спільноти. Результати включають тисячі учасників з відкритим вихідним кодом, веселі та інформативні конференції, величезна кількість дорогоцінних каменів (автономні рішення для певних проблем, таких як розбиття на сторінки та завантаження зображень), різноманітні інформативні блоги та рогу достатку дискусійних форумів Канали IRC. Велика кількість програмістів Rails також полегшує роботу з неминучими помилками додатків: алгоритм "повідомлення про помилку Google" майже завжди створює відповідне повідомлення в блозі або дискусійний форум.

Подібно до більшості мов програмування, Ruby має величезну кількість різноманітних бібліотек, які можуть допомогти розробникам на щоденній основі. Рубісти люблять використовувати каламбури, тому ми однозначно посилаємося на ці бібліотеки як на Gem. Кожен з цих gems містить окремі функції та індивідуальні методи, які можна викликати. Приклади функціональних можливостей можуть варіюватися від обробки платежів до обмеження доступу користувачів до тестового коду на наявність помилок. Вони часто працюють разом, щоб стати потужними інструментами при створенні програми або програми. Хоча більшість gem вважаються бібліотеками третіх сторін, і кожен може розміщувати свій власний приватний сервер gem, Ruby має величезну спільноту відкритих джерел gem. RubyGems.org - найпопулярніший хостинг для gem. Вона почалася в 2009 році як Gemcutter і підтримувалася партнерством по всій спільноті Ruby і через внески майже 300 рубістів. В даний час RubyGems.org приймає 146,000+ дорогоцінних каменів і нараховує понад 26 мільйонів завантажень gem. Будь-хто, хто хоче створити gem щоб допомогти спільноті або вирішити унікальну проблему, можна зробити так. Саме тому під час реалізації даного дипломного проекту було використано уже створені іншими користувачами gem’и. Їх дуже зручно імпортувати до проекту, необхідно додати

да файлу-конфігу Gemfile, та запустити команду, яка їх встановить. Gem aasm – створений для зручного управління статусами, він дає можливість швидко додавати статуси та дає зручний інтерфейс для їх управління. Gem bootstrap-sass підключає бібліотеку bootstrap до проекту, до дозволяє використовувати готові класи для стилізації веб-застосунку.

MySQL - система управління реляційними базами даних з відкритим вихідним кодом Oracle, що базується на Структурованій мові запитів (SQL). MySQL працює практично на всіх платформах, включаючи Linux, UNIX і Windows. Хоча MySQL може використовуватися в широкому діапазоні додатків, MySQL найчастіше асоціюється з веб-додатками та онлайн-публікаціями. MySQL є важливою складовою відкритого коду підприємства, що називається LAMP. LAMP - це платформа для веб-розробки, яка використовує Linux як операційну систему, Apache як веб-сервер, MySQL як реляційна система управління базами даних і PHP як об'єктно-орієнтована мова сценаріїв. (Іноді Perl або Python використовується замість PHP.) Спочатку задумана шведською компанією MySQL AB, MySQL була придбана компанією Sun Microsystems у 2008 році, а потім Oracle, коли вона придбала Sun в 2010 році. Розробники можуть використовувати MySQL за ліцензією GNU General Public License (GPL), але підприємства повинні отримати комерційну ліцензію Oracle. Сьогодні, MySQL є СУБД за багатьма провідними веб-сайтами у світі та незліченними корпоративними та споживчими веб-додатками, включаючи Facebook, Twitter і YouTube.

MySQL базується на моделі клієнт-сервер. Ядром MySQL є сервер MySQL, який обробляє всі інструкції (або команди) бази даних. Сервер MySQL доступний як окрема програма для використання в мережевому середовищі клієнт-сервер і як бібліотека, яка може бути вбудована (або пов'язана) в окремі програми. MySQL працює разом з декількома утилітами, які підтримують адміністрування баз даних MySQL. Команди надсилаються MySQLServer через клієнт MySQL, який встановлюється на комп'ютері. MySQL спочатку був

розроблений для швидкої обробки великих баз даних. Хоча MySQL, як правило, встановлюється тільки на одній машині, вона здатна посилати базу даних на декілька місць, оскільки користувачі мають доступ до неї через різні клієнтські інтерфейси MySQL. Ці інтерфейси надсилають SQL-оператори на сервер і потім відображають результати.

Інтерфейс користувача розроблений на мові програмування JavaScript з використанням CSS, HTML та веб-фреймворку Bootstrap. Компонент відображення використовується для всіх UI логіки програми. Наприклад, перегляд клієнта буде включати всі компоненти інтерфейсу користувача, такі як текстові поля, випадаючі списки тощо, з якими взаємодіє кінцевий користувач.

JavaScript є мовою програмування для Інтернету. Він підтримується більшістю веб-браузерів, включаючи Chrome, Firefox, Safari, Internet Explorer, Edge, Opera і т.д. Більшість мобільних браузерів для смартфонів також підтримують JavaScript. В основному він використовується для розширення веб-сторінок для забезпечення більш зручного для користувачів. До них відносяться динамічне оновлення веб-сторінок, удосконалення інтерфейсу користувача, такі як меню та діалогові вікна, анімація, 2D та 3D графіка, інтерактивні карти, відеопрогравачі та багато іншого. Цей режим використання JavaScript у веб-переглядачі також називається JavaScript на стороні клієнта.

Розглядаючи компоненти, які складають веб-сторінку, JavaScript формує третій компонент тріади, HTML і CSS, що є двома іншими. HTML описує сторінку, включаючи текст, графіку тощо. CSS використовується для керування та налаштування зовнішнього вигляду веб-сторінки, включаючи кольори, шрифти та ін. елементи на сторінці програмуються (Рисунок 2.1.2).

Jquery - це бібліотека відкритих кодів JavaScript, яка спрощує взаємодію між документом HTML / CSS, точніше, об'єктною моделлю документа (DOM) і JavaScript. Розробляючи терміни, jquery спрощує проходження та маніпулювання HTML-документами, обробку подій у браузері, анімацію DOM, взаємодії Ajax і розробку крос-браузерних JavaScript.

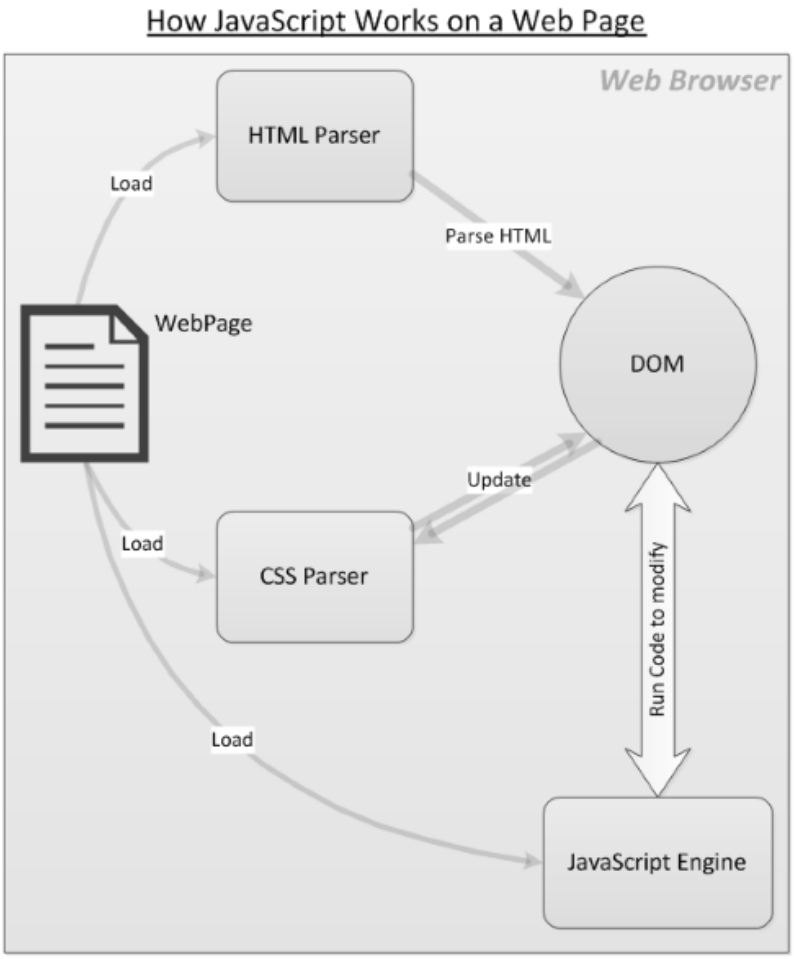

Рисунок 2.1.2 – Схематичне зображення взаємодії компонентів клієнту[15]

Деякі ключові моменти, які підтримують відповідь, чому використовувати jQuery:

– це неймовірно популярна, тобто вона має велику спільноту користувачів, які беруть участь як розробники;

– він нормалізує відмінності між веб-переглядачами, так що вам не потрібно;

– це навмисно легкий слід з простою, але розумною архітектурою плагінів.

– його репозиторій плагінів величезний і спостерігається стійке зростання з моменту випуску jQuery;

– його API повністю задокументований, включаючи приклади вбудованого коду, які в світі бібліотек JavaScript є розкішшю. Чорт, будь-яка документація взагалі була розкішшю на роки;

– вона зручна, тобто вона надає корисні способи уникнення конфліктів з іншими бібліотеками JavaScript.

HTML – це комп'ютерна мова, створена для створення веб-сайтів. Ці веб-сайти можуть переглядати будь-хто, хто підключився до Інтернету. Це відносно легко навчитися, з основами, доступними для більшості людей за один прийом; і досить потужний у тому, що він дозволяє вам створювати. Вона постійно проходить ревізію та еволюцію, щоб відповідати вимогам та вимогам зростаючої інтернет-аудиторії під керівництвом організації W3C, організації, яка зобов'язана проектувати та підтримувати мову.

HTML складається з серії коротких кодів, введених автором сайту в текстовий файл – це теги. Потім текст зберігається як html-файл і переглядається через браузер, наприклад Internet Explorer або Netscape Navigator. Цей веб-переглядач читає файл і переводить текст у видиму форму, сподіваючись надати цю сторінку, як передбачав автор. Написання власного HTML-коду означає правильне використання тегів для створення свого бачення. Ви можете використовувати що-небудь від рудиментарного текстового редактора до потужного графічного редактора для створення HTML-сторінок.

Каскадні таблиці стилів, що називають CSS, є простою мовою дизайну, призначеної для спрощення процесу створення веб-сторінок. CSS обробляє зовнішній вигляд веб-сторінки. Використовуючи CSS, можна керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром та розміщенням стовпців, які фонові зображення або кольори використовуються, макетом, варіантами відображення для різних пристроїв і розмірів екрана а також безліч інших ефектів. CSS легко вчитися і розуміти, але він забезпечує потужний контроль над презентацією HTML-документа. Найчастіше, CSS поєднується з мовами розмітки HTML або XHTML. CSS створюється і підтримується через

групу людей у W3C під назвою CSS Working Group. Робоча група CSS створює документи, які називаються специфікаціями. Коли специфікація обговорюється та офіційно ратифікована членами W3C, вона стає рекомендацією. Ці ратифіковані специфікації називаються рекомендаціями, оскільки W3C не контролює фактичну реалізацію мови. Незалежні компанії та організації створюють це програмне забезпечення. Переваги CSS:

– CSS економить час - можна писати CSS один раз, а потім повторно використовувати той самий аркуш на декількох сторінках HTML. Ви можете визначити стиль для кожного елемента HTML і застосувати його до якомога більшої кількості веб-сторінок;

– сторінки завантажуються швидше. Якщо ви використовуєте CSS, вам не потрібно кожного разу писати атрибути тегів HTML. Просто напишіть одне правило CSS тегу і застосуйте його до всіх входжень цього тегу. Тому менше коду означає швидше завантаження;

– легке технічне обслуговування - Щоб зробити глобальні зміни, просто змініть стиль, і всі елементи на всіх веб-сторінках будуть автоматично оновлюватися;

– різні стилі для HTML - CSS має набагато ширший масив атрибутів, ніж HTML, так що ви можете набагато краще поглянути на вашу HTML-сторінку порівняно з атрибутами HTML;

– сумісність з кількома пристроями - таблиці стилів дозволяють оптимізувати вміст для більш ніж одного типу пристроїв. Використовуючи один і той же документ HTML, для портативних пристроїв, таких як КПК і стільникові телефони або для друку, можуть бути представлені різні версії веб-сайту;

– глобальні веб-стандарти - тепер атрибути HTML застаріли, і рекомендується використовувати CSS. Таким чином, це гарна ідея, щоб почати використовувати CSS на всіх HTML-сторінках, щоб зробити їх сумісними з майбутніми браузерами.

Bootstrap – це вільний і відкритий код для створення веб-сайтів і веб-додатків. Це найбільш популярний HTML, CSS і JS-фреймворк для розробки адаптивних, мобільних перших проектів в Інтернеті. Оскільки веб-сайти все більше розвиваються до чуйного дизайну, це може бути реальним викликом для веб-розробників. Bootstrap може зробити речі набагато простіше. Bootstrap дозволяє створювати відповідні веб-сайти без необхідності виконувати біт "responsive". Bootstrap піклується про це. Bootstap включає такі компоненти, як кнопки, навігаційні панелі, випадаючі меню, вікна сповіщення та багато іншого. У більшості випадків ви можете використовувати компонент просто за допомогою відповідного імені класу. Однією з головних переваг фреймворків розробки, таких як Bootstrap, є те, що вони можуть допомогти прискорити час розробки, зберігаючи при цьому якість і узгодженість на сайті. Вам більше не потрібно повторно проектувати кожен елемент. І вам не потрібно витрачати годинник, намагаючись отримати все, що виглядає і працює прямо в браузерах, платформах і пристроях. Використовуючи Bootstrap, все (більшість) важкої роботи є для вас.

Враховуючи, що Bootstrap є найпопулярнішим фреймворком для розробки веб-інтерфейсів, робота з цим фреймворком є корисною для вивчення та використовування у проекті. Додавання Bootstrap до веб-застосування може допомогти вам у багатьох відношеннях – від створення веб-сайтів швидше, та створювати дизайн для зручного користування без зусиль.

У таблиці 2.1 наведено класи та їх опис.

Таблиця 2.1 – Опис класів (структур) системи

Клас (структура)	**Опис**
ApplicationController	Клас, який містить загальну бізнес-логіку системи.
PagesController	Клас, який містить бізнес-логіку відображення сторінок .

Продовження до таблиці 2.1

Клас (структура)	**Опис**
PetitionsController	Клас, який містить бізнес-логіку компоненту петицій .
SessionsController	Клас, який містить бізнес-логіку для роботи сесій.
UsersController	Клас, який містить бізнес-логіку компоненту користувачів.
Petition	Клас, який містить логічні данні компоненту петицій.
User	Клас, який містить логічні данні компоненту користувач.
Routes	Клас, який містить бізнес логіку для маршрутизації системи.
Schema	Клас, який містить поточні представлення бази даних.
PetitionsControllerTest	Клас, який містить тест-кейси для компоненту петицій.
SessionsControllerTest	Клас, який містить тест-кейси для компоненту сесій.
UsersControllerTest	Клас, який містить тест-кейси для компоненту користувача.
CreateUserTest	Клас, який містить тест-кейси для створення користувача.
PetitionTest	Клас, який містить тест-кейси для створення петиції.
UserTest	Клас, який містить тест-кейси для створення користувача.

Продовження до таблиці 2.1

Клас (структура)	**Опис**
HardWorker	Клас, який містить бізнес-логіку компоненту робітник.
HardWorkerTest	Клас, який містить тест-кейси компоненту робітник.
PagesIndex	Клас, який містить відображення головної сторінки системи.
PetitionIndex	Клас, який містить відображення головної сторінки компоненту петицій.
PetitionNew	Клас, який містить відображення сторінки створення петиції компоненту пецій.
PetitionShow	Клас, який містить відображення сторінки конкретної петиції компоненту пецій.
UsersNew	Клас, який містить відображення сторінки створення користувача компоненту користувач.
UsersShow	Клас, який містить відображення сторінки конкретного користувача компоненту користувач.
SessionsNew	Клас, який містить відображення сторінки створення сесії компоненту сесії.
ApplicationMailer	Клас, який містить бізнес-логіку відправки електронних повідомлень.

У таблиці 2.2 представлено класи і опис методів, що реалізовано.

Таблиця 2.2 – Опис методів класів та інтерфейсів системи

Клас/Інтерфейс	**Метод**	**Опис**
ApplicationController	current_user ()	Повертає користувача, який авторизований в системі.
ApplicationController	logged_in ()	Перевіряє чи авторизований користувач.
ApplicationController	require_user()	Вимагає щоб користувач був авторизований.
PagesController	index()	Повертає всі петиції з бази даних.
PetitionsController	index()	Перевірка всіх петицій та розподіл їх по сторінках.
PetitionsController	new()	Створення нової петиції.
PetitionsController	create(title, description, user_id)	Створення нової петиції з параметрами.
PetitionsController	destroy()	Видалення петиції.
PetitionsController	support()	Додання до кількості голосів петиції голосу, якщо її підтримали.

Продовження до таблиці 2.2

Клас/Інтерфейс	**Метод**	**Опис**
PetitionsController	unsupport()	Віднімання від кількості голосів петиції голосу, якщо користувач передумав.
PetitionsController	show()	Показ петиції.
PetitionsController	set_article()	Пошук петиції в базі даних.
SessionsController	create ()	Створює сесію користувача в системі..
SessionsController	destroy()	Знищує поточної сесії користувача в системі.
UsersController	create(username, email, password, first_name, second_name, city, passport_number, phone_number, date_of_birth)	Створює нового користувача.
UsersController	edit(user_id)	Редагує дані користувача.
Petition	check_voute_size()	Перевіряє кількість голосів підтримки петиції.
Petition	complete()	Змінює статус петиції з «очікує відповіді» до «закрита».

Продовження до таблиці 2.2

Клас/Інтерфейс	**Метод**	**Опис**
Petition	wait()	Змінює статус петиції з «на голосуванні» до «очікує відповіді».
Petition	decline()	Змінює статус петиції з «на голосуванні» до «відмінена».
Schema	create_table()	Створює таблицю в базі даних.

2.2 Аналіз безпеки даних

Фреймворки веб-додатків створені, щоб допомогти розробникам створювати веб-програми. Деякі з них також допомагають захистити веб-додаток. Насправді один фреймворк не є більш безпечним, ніж інший: якщо ви використовуєте його правильно, ви зможете створювати безпечні програми з багатьма фреймворками. Ruby on Rails має деякі розумні допоміжні методи, наприклад, проти ін'єкції SQL, так що це навряд чи є проблемою.

Взагалі немає такої речі, як безпека фреймворків. Безпека залежить від людей, які використовують фреймворки, а іноді і від методу розробки. І це залежить від всіх шарів середовища веб-додатків: базового сховища, веб-сервера і самого веб-додатка (і, можливо, інших шарів або додатків).

Група Gartner, однак, оцінює, що 75% атак знаходяться на рівні веб-додатків, і виявив, що «з 300 перевірених сайтів, 97% є вразливими до атак». Це пояснюється тим, що веб-додатки відносно легко атакувати, оскільки вони прості для розуміння і маніпулювання, навіть мирянином.

Загрози для веб-додатків включають захоплення облікового запису користувача, обхід контролю доступу, читання чи зміну конфіденційних даних

або представлення шахрайського вмісту. Або зловмисник може встановити програму троянського коня або програмне забезпечення для надсилання електронної пошти, спрямоване на фінансове збагачення або призвести до пошкодження назви бренда шляхом зміни ресурсів компанії. Щоб запобігти атакам, мінімізувати їх вплив і видалити точки атаки, перш за все, ви повинні повністю зрозуміти методи атаки, щоб знайти правильні контрзаходи.

Більшість програм потребує відстеження певного стану певного користувача. Це може бути вміст кошика для покупок або ідентифікатор користувача поточного зареєстрованого користувача. Без ідеї сеансів, користувачеві доведеться ідентифікувати і, можливо, автентифікувати, при кожному запиті. Rails створить новий сеанс автоматично, якщо новий користувач звертається до програми. Він завантажуватиме існуючий сеанс, якщо користувач вже використовував цю програму. Сесія зазвичай складається з хешу значень і ідентифікатора сеансу, зазвичай 32-символьного рядка, для ідентифікації хешу. Кожен файл cookie, надісланий до браузера клієнта, містить ідентифікатор сеансу. І навпаки: браузер буде відправляти його на сервер при кожному запиті від клієнта. У Rails можна зберігати та отримувати значення за допомогою методу сеансу.

Багато веб-додатків мають систему автентифікації: користувач надає ім'я користувача і пароль, веб-додаток перевіряє їх і зберігає відповідний ідентифікатор користувача в сеансі хешу. Відтепер сесія є дійсною. При кожному запиті програма завантажує користувача, ідентифікованого ідентифікатором користувача в сесії, без необхідності нової автентифікації. Ідентифікатор сеансу у файлі cookie ідентифікує сеанс. Таким чином, cookie служить тимчасовою автентифікацією для веб-програми. Той, хто скористається файлом cookie від когось іншого, може використовувати веб-додаток як цей користувач - з можливими серйозними наслідками.

Також основною проблемою є зберігання даних користувача в базі даних на сервері. Хакери можуть отримати доступ до даних користувача, та здійснити віх

до його кабінету. Система збору та обробки електронних петицій повинна ідентифікувати користувача за номером паспорту тому, що не повинно буди фальсифікації голосування за петиції та їх створення.

Саме тому паролі користувачів та номери їх паспортів у базі зберігаються у вигляді результату обробки функцією bcrypt, тому праобраз паролю майже неможливо знайти без повного перебору, який буде сповільнено через особливість роботи bcrypt що змушує витрачати багато часу на визначення хешу паролю.

Висновок до розділу

У даному розділі було розроблено архітектуру системи.

Обрано мову, яка найкраще підходить для реалізації поставленої задачі Ruby on Rails так як розробка на Ruby on Rails дуже швидка, має готові сервіси для тестування, є вже готова валідація, робота з роутингом, зручна робота з базами даних та їх міграції та головне безпека, обрано паттерн на основі якого розроблено систему.

Обрано мову для написання веб-інтерфейсу, Javascript в поєднані з HTML та CSS.

Описано інтерфейси, класи, структури даних комплексу та описано їх методи. Проаналізовано безпеку даних у комплексі.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування програмного забезпечення – це оцінка програмного забезпечення щодо вимог, зібраних користувачами та специфікаціями системи. Тестування проводиться на фазовому рівні в життєвому циклі розробки програмного забезпечення або на рівні модуля в програмному коді.

Тестування можна виконати вручну або за допомогою автоматизованого засобу тестування:

– ручне – це тестування виконується без використання автоматизованих засобів тестування. Тестер програмного забезпечення готує тестові приклади для різних розділів і рівнів коду, виконує тести і повідомляє менеджеру результат. Ручне тестування є витратним часом та ресурсами. Тестер повинен підтвердити, чи використовуються правильні тести. Основна частина тестування включає ручне тестування;

– автоматизоване – це тестування є процедурою тестування, що виконується за допомогою автоматизованих засобів тестування. Обмеження з ручним тестуванням можна подолати за допомогою автоматизованих тестових інструментів.

Тестер повинен перевіряти, чи можна відкрити веб-сторінку в Internet Explorer. Це можна легко зробити за допомогою ручного тестування. Але перевірити, чи може веб-сервер зайняти навантаження 1 мільйон користувачів, тестування вручну неможливо. Існують програмні та апаратні засоби, які допомагають тестеру у проведенні навантажувального тестування, стрес-тестування, регресійного тестування. Тести можуть проводитися на основі двох підходів:

– функціональне тестування;

– тестування впровадження.

Коли функціональність тестується, не беручи до уваги фактичну реалізацію, вона називається тестуванням на чорному ящику. Інша сторона відома як тестування білих коробок, де не тільки тестується функціональність, але й аналізується спосіб її реалізації. Вичерпні тести є найкращим методом для ідеального тестування. Кожне можливе значення в діапазоні вхідних і вихідних значень перевіряється. Неможливо протестувати кожне значення в реальному світі, якщо діапазон значень великий.

Тестування в чорному ящику проводиться для перевірки функціональності програми. Його також називають «поведінковим» тестуванням. Тестер в цьому випадку має набір вхідних значень і відповідних бажаних результатів. При введенні даних, якщо вихідний результат відповідає бажаним результатам, програма перевіряється «нормально», і в іншому випадку проблематично. У цьому методі тестування, тестування не відомі конструкції і структурі коду, а тестуючі інженери і кінцеві користувачі проводять цей тест на програмному забезпеченні(Рисунок 3.1.1).

Рис. 3.1.1 Схема тестування в чорному ящику[3]

Методи тестування в чорному ящику:

– клас еквівалентності – вхід поділяється на подібні класи. Якщо один елемент класу проходить тест, передбачається, що весь клас передається;

– граничні значення – вхід розділений на більш високі і нижчі кінцеві значення. Якщо ці значення проходять тест, то передбачається, що всі значення між ними також можуть пройти;

– графік причинно-наслідкового впливу – в обох попередніх методах перевіряється тільки одне вхідне значення. Причина (вхід) – Ефект (висновок) – це метод тестування, в якому систематично перевіряються комбінації вхідних значень;

– парне тестування – поведінка програмного забезпечення залежить від декількох параметрів. При попарному тестуванні множинні параметри випробовуються для різних значень;

Тестування в білому ящику проводиться для тестування програми та її реалізації, з метою підвищення ефективності коду або структури. Він також відомий як «Структурне» тестування.

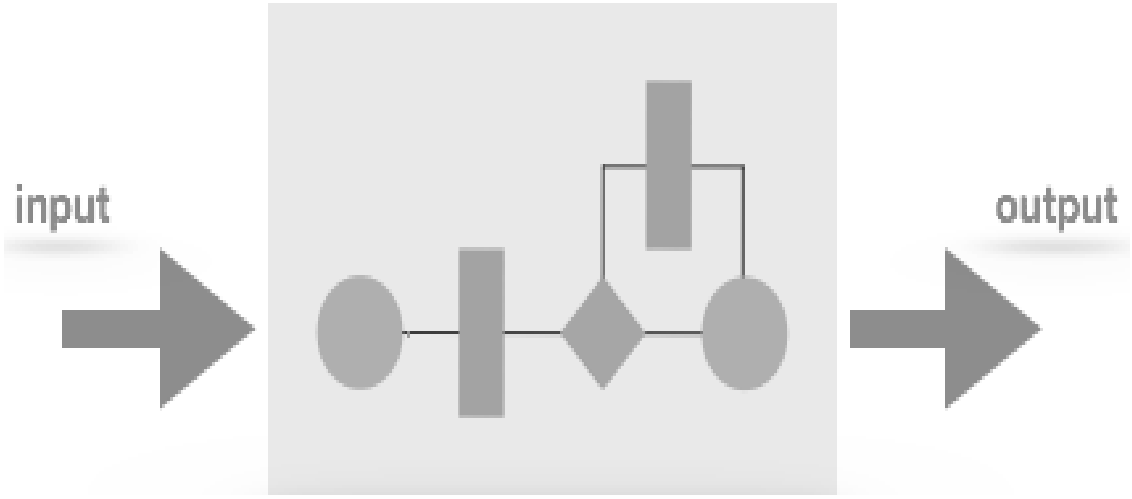

Рис. 3.1.2 Схема тестування в білому ящику [3]

У цьому методі тестування, тестування відомі конструкція і структура коду. Програмісти коду проводять цей тест на коді.

Контрольно-потокове тестування – мета тестування контрольного потоку для встановлення тестових випадків, що охоплюють всі твердження та гілки. Умови гілок тестуються для істинного і неправдивого, так що всі заяви можуть бути покриті.

Тестування потоку даних – дана тестова техніка акцентує увагу на всіх змінних даних, включених в програму. Вона перевіряє, де змінні були оголошені і визначені і де вони використовувалися або змінювалися.

Етап тестування є важливим в процесі розробки даного продукту у зв`язку з великою складністю системи. Також продукт передбачає постійну взаємодію із зовнішніми онлайн сервісами.

Саме тестування може бути визначено на різних рівнях SDLC. Процес тестування йде паралельно з розробкою програмного забезпечення. Перед тим, як перейти на наступний етап, етап перевіряється, перевіряється і перевіряється. Тестування окремо робиться тільки для того, щоб переконатися, що в програмному забезпеченні немає прихованих помилок або проблем. Програмне забезпечення тестується на різних рівнях.

Тестування одиниць – під час кодування програміст виконує деякі тести на цій одиниці програми, щоб дізнатися, чи не є помилкою. Тестування проводиться за методом білого ящика. Блок тестування допомагає розробникам вирішити, що окремі одиниці програми працюють за вимогами і не допускають помилок.

Тестування інтеграції – навіть якщо одиниці програмного забезпечення працюють відмінно індивідуально, необхідно з'ясувати, чи будуть одиниці, якщо вони об'єднані разом, працювати без помилок. Наприклад, передача аргументів і оновлення даних і т.д.

Тестування системи – програмне забезпечення компілюється як продукт, а потім випробовується в цілому. Це може бути виконано за допомогою одного або декількох з наступних тестів:

– функціональне тестування – перевіряє всі функціональні можливості програмного забезпечення на вимогу;

– тестування продуктивності – цей тест підтверджує ефективність програмного забезпечення. Вона перевіряє ефективність і середній час, необхідний програмному забезпеченню для виконання бажаної задачі. Тестування продуктивності здійснюється за допомогою тестування

навантаження та стрес-тестування, де програмне забезпечення піддається високому завантаженню користувачів і даних при різних умовах навколишнього середовища;

– безпека та портативність – ці тести виконуються, коли програмне забезпечення призначене для роботи на різних платформах і доступ до яких здійснюється за кількістю осіб.

Приймальна перевірка – коли програмне забезпечення готове передати клієнтові, воно має пройти останній етап тестування, де він протестований для взаємодії з користувачем та відповіді. Це важливо, оскільки навіть якщо програмне забезпечення відповідає всім вимогам користувача, і якщо користувачеві не подобається те, як він з'являється або працює, його можна відхилити.

Альфа-тестування – команда розробників виконує альфа-тестування за допомогою системи, як якщо б вона використовувалася в робочому середовищі. Вони намагаються з'ясувати, як користувач реагує на певні дії в програмному забезпеченні і як система повинна реагувати на вхідні дані.

Бета-тестування – після того, як програмне забезпечення перевірено внутрішньо, воно передається користувачам, щоб використовувати його у виробничому середовищі тільки для цілей тестування. Це ще не доставлений продукт. Розробники очікують, що користувачі на цьому етапі принесуть хвилинні проблеми, які були пропущені для участі.

Тестування регресії – всякий раз, коли програмний продукт оновлюється новим кодом, функцією або функціональністю, він ретельно перевіряється для виявлення негативного впливу доданого коду. Це відоме як регресійне тестування.

План тестування ПЗ включає в себе обсяг, підхід, ресурси та план усіх методів тестування. План описує об’єкти та функції що будуть протестовані, тип тестів, ресурси та план необхідний для виконання тестування.

У рамках цього плану буде виконано тестування частини продукту, що відповідає за процес інтегрування програмного коду продукту.

У даному плані будуть протестовані наступні функції:

- реєстрація користувачів;
- авторизація користувачів;
- створення петицій;
- голосування за петицій;
- відправка петицій;
- змінення статусу петицій;
- видалення петицій;
- зміна даних користувачів;
- архівування петицій.

Налаштовані наступні тестові модулі:

- реєстрація користувача;
- реєстрація користувача якщо логін вже зайнято;
- реєстрація користувача з невірним паролем;
- авторизація користувача;
- авторизація користувача з неправильним паролем;
- створення петицій;
- створення петицій з замалою кількістю символів;
- голосування за петиції;
- відправка петицій;
- відправка петицій, якщо кількість голосів замала;
- відправка петицій, якщо статус змінено;
- змінення статусу петицій;
- змінення статусу петицій на неіснуючий;
- видалення петицій;
- видалення не існуючих петицій;

– зміна даних користувачів;
– зміна даних користувачів на правильні дані;
– архівування петицій.

3.2 Підходи до тестування

В рамках даного плану будуть використані наступні методи тестування:
– продуктивності;
– інтеграційне.

3.2.1 Інтеграційне тестування

Інтеграційне тестування – це рівень тестування програмного забезпечення, де окремі одиниці об'єднуються і перевіряються як група (Рисунок 3.2.1).

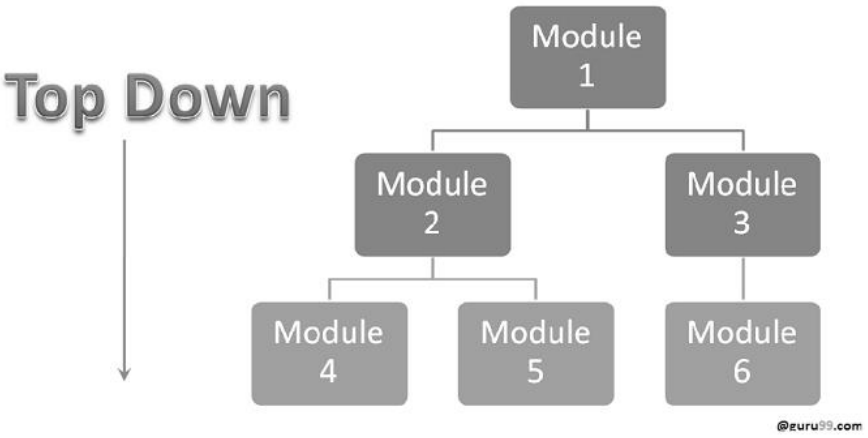

Рисунок 3.2.1 – Схематичне зображення принципу інтеграційного тестування[14]

Метою цього рівня тестування є виявлення несправностей у взаємодії між інтегрованими блоками. Для допомоги в тестуванні інтеграції використовуються тестові драйвери та тестові заглушки.

Методом інтеграційного тестування будуть перевірені взаємодії між

модулями системи, такі як робота сервера з базою даних.

3.2.2 Тестування продуктивності

Тестування продуктивності виконується для забезпечення зацікавлених сторін інформацією про їх застосування щодо швидкості, стабільності та масштабованості. Більш важливим є те, що тестування продуктивності виявляє те, що потрібно покращити до того, як продукт потрапить на ринок. Без тестування продуктивності, програмне забезпечення, швидше за все, страждають від таких питань, як: працює повільно, в той час як кілька користувачів використовують його одночасно, невідповідності в різних операційних системах і поганий юзабіліті.

Методом тестування продуктивності буде перевірена швидкодія наступного елементу системи – запити до бази даних.

3.3 Критерії проходження тестування

3.3.1 Інтеграційне тестування

Для інтеграційного тестування критерієм проходження є успішне виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим.

3.3.2 Тестування швидкодії

Для тестування швидкодії критерієм проходження є успішне виконання тесту з кожним доступним набором параметрів (кількість даних у одному запиті, кількість запитів, конкретність) не довше ніж максимально допустимий час. У разі якщо хоча б один варіант тесту не був успішним або виконувався довше

максимально допустимого часу – тестування вважається не пройденим.

3.4 Процес тестування

3.4.1 Дані до тестів

Вхідними даними для компонентного тестування є набори параметрів на яких очікується певний результат, що є вихідними даними даного тесту.

Вхідними даними для інтеграційного тестування є набори повідомлень що будуть передані від одного компоненту системи до іншого відповідно до конкретного тесту. Вихідними даними для даного виду тестування є результат роботи останнього компоненту у ланцюзі (наприклад, запис у базі даних у випадку тестування взаємодії серверу та бази даних).

Вхідними даними до тестування швидкодії є набори даних, що покривають усі варіанти роботи системи у конкретному випадку. Вихідними даними є швидкість обробки запитів, кількість оброблених запитів, кількість неправильних реакцій на набір даних, дані по навантаженню на апаратну платформу (завантаженість процесору, вільна оперативна пам’ять, завантаженість мережі тощо).

3.4.2 Задачі тесту

Кожен тест потрібен, щоб перевірити роботу застосунку без помилок, або виявити помилки на момент розробки, та полагодити роботу застосування.

3.4.3 План виконання

Інтеграційного тестування виконується методом чорної скриньки та повинне виконуватися перед виконання тестування швидкодії.

3.5 Вимоги до середовища

3.5.1 Апаратна частина

- тип процесору x86_64 або ARM v8 64;
- об‘єм ОЗП 1024 Мб.;
- 1 Гб постійного дискового простору;
- клавіатура.

3.5.2 Інструменти

Для виконання тестування використовувати наступні програмні інструменти:

- RSpec;
- Vegeta (https://github.com/tsenart/vegeta) – для тестування швидкодії.

3.6 Висновок до розділу

У даному розділі було обрані та проаналізовані підходи до тестування системи, визначено критерії для проходження різних типів тестувань. Описано процес тестування, а саме:

- вхідні дані до тестів;
- задачі тестування;
- план виконання тестування.

Було проаналізовано та виділено вимоги до середовища, а саме мінімальні системні вимоги, для успішної роботи системи.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для повного розгортання даного комплексу потрібно виконати наступні етапи:

- налаштування середи Ruby on Rails;
- встановлення Phusion Passenger;
- налаштувати Ngnix;
- налаштування Nginx для розгортання програми;
- завантаження і установка Capistrano.

4.1.1 Налаштування середи Ruby on Rails

Для успішної роботи системи її необхідно розгорнути на сервері, щоб користувачі мали доступ до системи через клієнт, та могли взаємодіяти з системою.

Необхідно запустити наступні команди, щоб встановити RVM і створити середовище для Ruby:

- curl -L get.rvm.io | bash -s stable source /etc/profile.d/rvm.sh;
- rvm reload;
- rvm install 2.1.0.

Rails потрібен інтерпретатор JavaScript, встановіть для цього Node.js. Необхідно запустити наступну команду, щоб завантажити і встановити nodejs за допомогою yum:

- yum install -y nodejs

Додамо gem для Rails:

- gem install bundler rails

4.1.2 Встановлення Phusion Passenger

Phusion Passenger - це безкоштовний веб-сервер і сервер додатків з підтримкою Ruby, Python і Node.js. Він призначений для інтеграції в Apache HTTP Server або веб-сервер nginx, але також має режим для роботи автономно без зовнішнього веб-сервера.

Менеджер пакетів Red Hat за замовчуванням RPM (RPM Package Manager) надає додатки у файлах .rpm. На жаль, файли для Phusion Passenger досить сильно застаріли. Тому потрібно використовувати RubyGem, щоб завантажити і встановити останню доступну версію Passenger.

– gem install passenger

4.1.3 Встановлення Ngnix

Ngnix - це програмне забезпечення з відкритим вихідним кодом для веб-сервісу, зворотного проксі, кешування, вирівнювання навантаження, потокового передавання медіа та багато іншого. Вона почалася як веб-сервер, призначений для максимальної продуктивності та стабільності. На додаток до можливостей HTTP-сервера, NGINX також може працювати як проксі-сервер для електронної пошти (IMAP, POP3 і SMTP), а також зворотного проксі-сервера і балансування навантаження для серверів HTTP, TCP і UDP.

Зазвичай для установок і налаштувань використовується Nginx репозиторій EPEL. Для того, щоб сервер Nginx міг працювати з Пасажиром, його вихідний код потрібно скомпілювати з додатковими модулями. Необхідно запустити наступну команду, щоб скомпілювати Nginx з модулем Passenger:

– passenger-install-nginx-module

Після запуску команди необхідно натиснути Enter і вибрати (в даному випадку Ruby). Щоб вибрати Ruby, за допомогою клавіш зі стрілками і пробіл.

Необхідно натиснути Enter, щоб продовжити. Після цього коду Nginx буде завантажено и встановлено з підтримкою Passenger.

4.1.4 Налаштування Nginx для розгортання програми

На цьому останньому етапі налаштування серверів потрібно створити віртуальний хост (блок server) Nginx. Для цього потрібно додати блок коду в конфігураційний файл Nginx, nginx.conf. За замовчуванням цей файл можна знайти в /opt/nginx/conf/nginx.conf. Щоб відкрити конфігураційний файл, потрібно запустити наступну команду:

– nano /opt/nginx/conf/nginx.conf

Необхідно знайти блок http і вставити такі рядки після директив passenger_root і passenger_ruby:

– passenger_app_env development;

Потім потрібно знайти блок server, закоментувати location за замовчуванням та визначити корінний каталог додатку і зберегти файл.

Необхідно запустити наступну команду, щоб перезапустити Nginx і оновити налаштування:

– /etc/init.d/nginx restart

Далі необхідно перевірити стан Nginx:

– /etc/init.d/nginx status

4.1.5 Завантаження і установка Capistrano

Тепер система готова. Потрібно встановити останню версію Capistrano з RubyGems.

– gem install capistrano

Щоб почати використовувати Capistrano, потрібно встановити бібліотеку всередині каталогу проекту. На цьому етапі потрібно додати Capistrano в каталог проекту і створити файли, необхідні для налаштування серверів.

4.2 Інструкція користувача

4.2.1 Початок роботи з системою

Для роботи з веб додатком користувач спершу потрапляє на головний екран додатку, який дає зрозуміти, що це за додаток та користувач може вибирати свій наступний крок.(Рисунок 4.2.1).

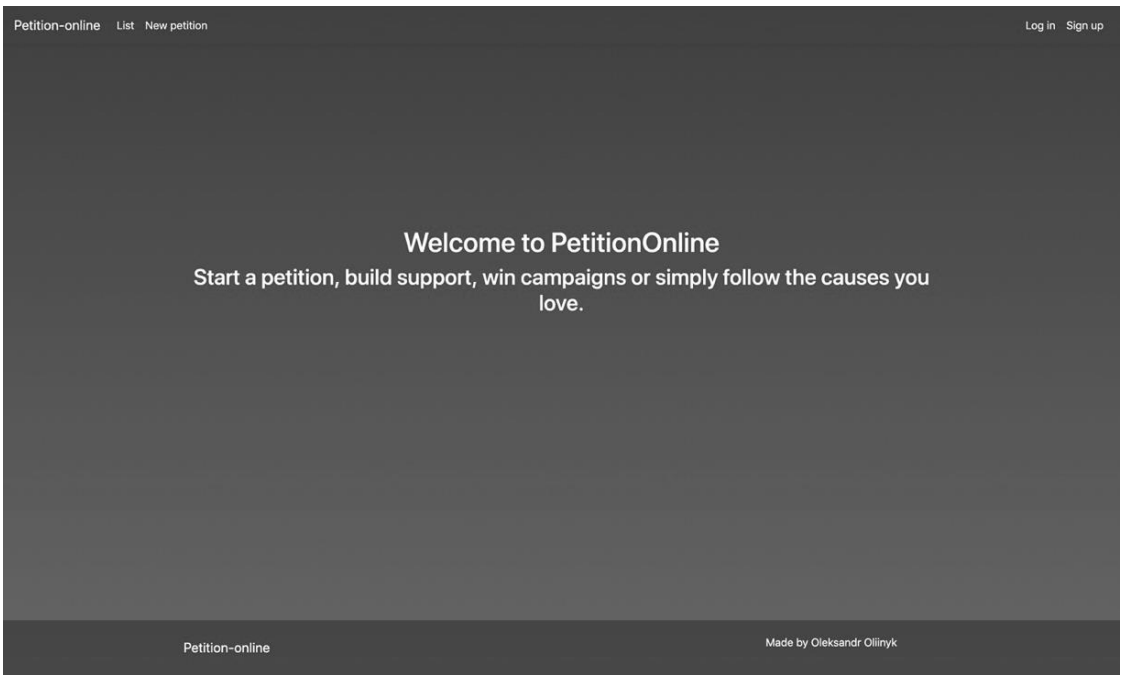

Рисунок 4.2.1 – Головна сторінка

4.2.2 Реєстрація в системі

Для початку роботи з веб додатком, користувач повинен бути зареєстрований у системі. Якщо цього не зробити, доступ та користування деякими частинами додатку буде обмежений. Для успішної реєстрації потрібно вести ім’я

користувача в системі, email адресу, ім’я, прізвище, пароль, місто проживання, номер паспорту, номер телефону та дату народження(Рисунок 4.2.2).

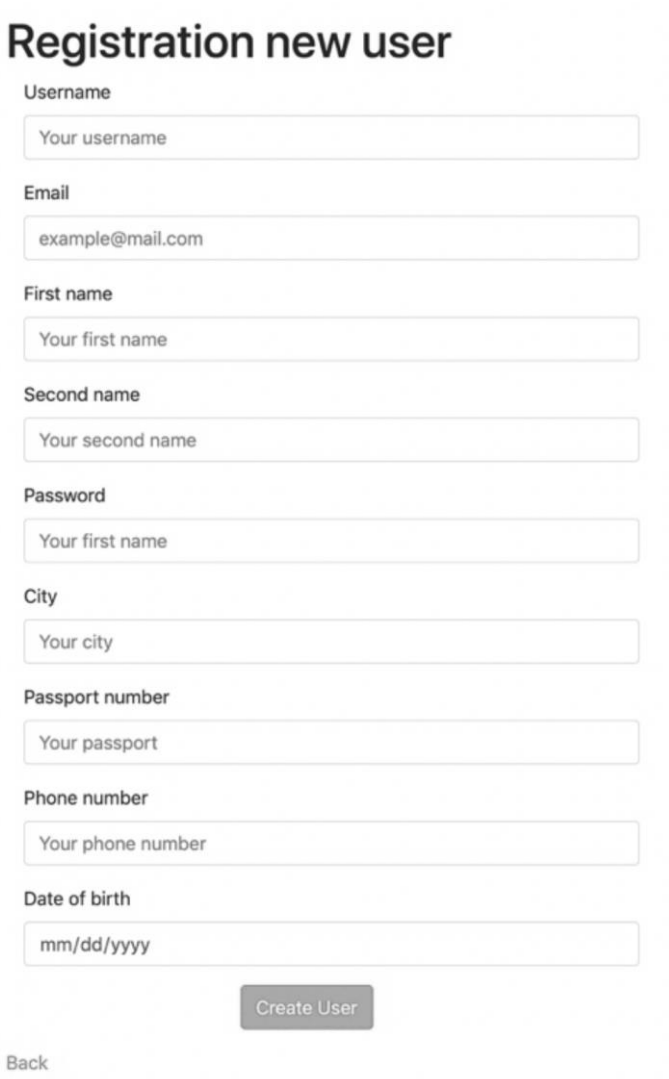

Рисунок 4.2.2 – Сторінка реєстрації нового користувача

Користувач також може ввести не вірні данні, тоді система виведе користувачу де і що саме було введено не вірно (Рисунок 4.2.3).

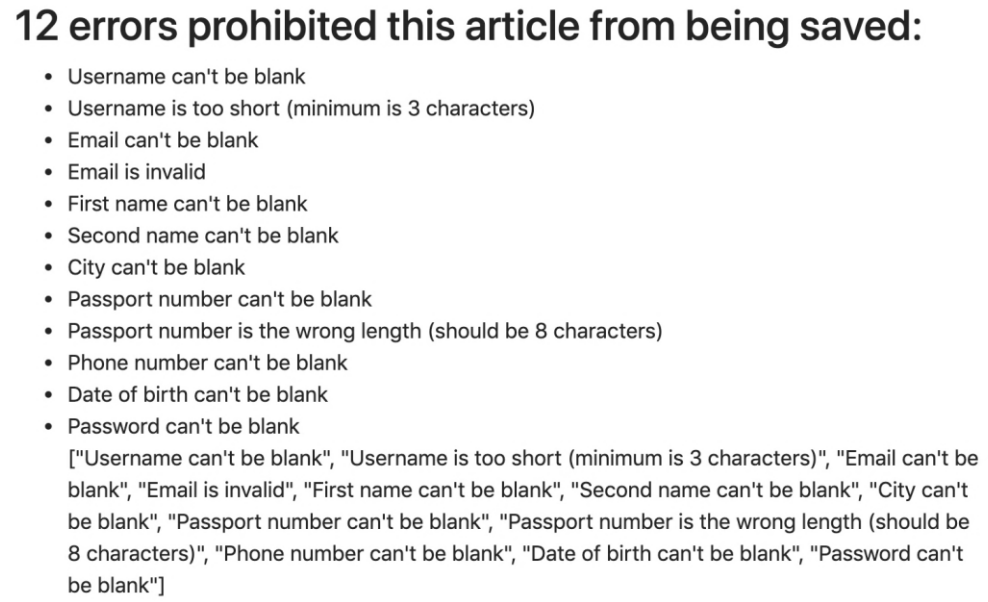

Рисунок 4.2.3 – Помилка при реєстрації нового користувача

4.2.3 Логін в системі

Якщо користувач вже має свій персональний акаунт в системі, він може виконати логін в системі. Тільки коли користувач увійшов в систему за допомогою email та паролю він отримує доступ до повного функціонування системи без обмежень. Для цього необхідно ввести email та пароль в поля форми (Рисунок 4.2.4).

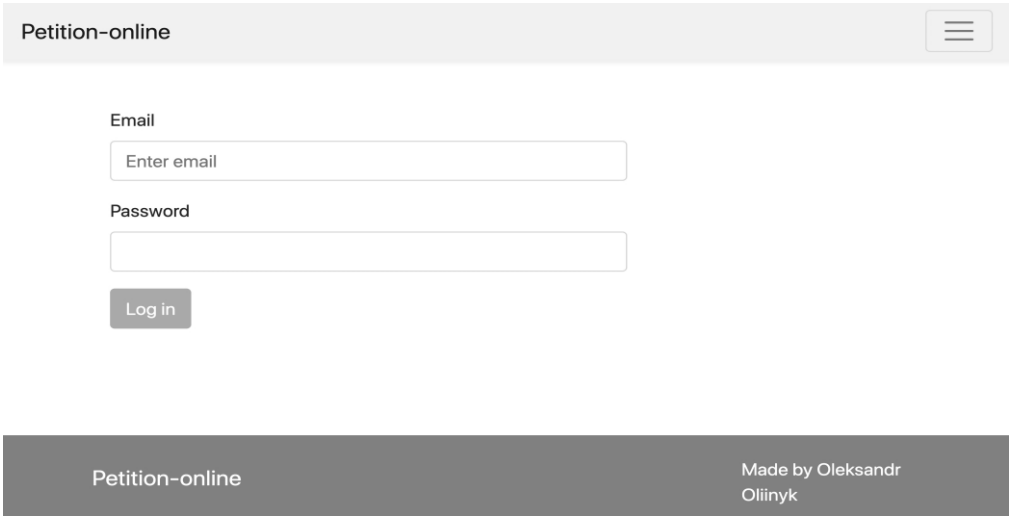

Рисунок 4.2.4 – Сторінка логіну в систему

При введені помилкових даних, яких немає в системі, користувач отримає повідомлення (Рисунок 4.2.5).

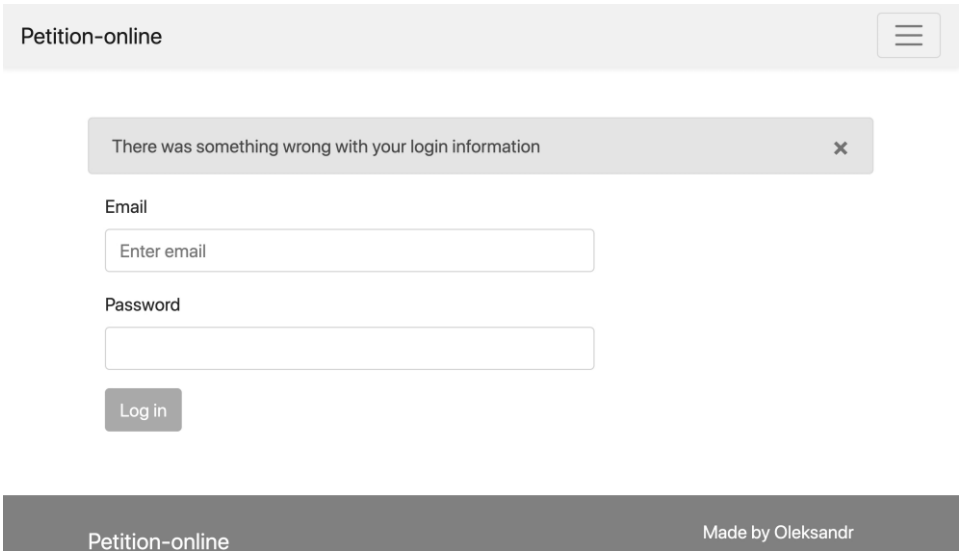

Рисунок 4.2.5 – Сторінка не успішного логіну в систему

4.2.4 Створення петиції

Тепер, коли користувач увійшов до системи він отримує доступ до створення нової петиції. Необхідно перейти на сторінку створення петиції. Для створення петиції необхідно ввести в форму Короткий заголовок, Організацію отримувача петиції та Описати суть петиції(Рисунок 4.2.6).

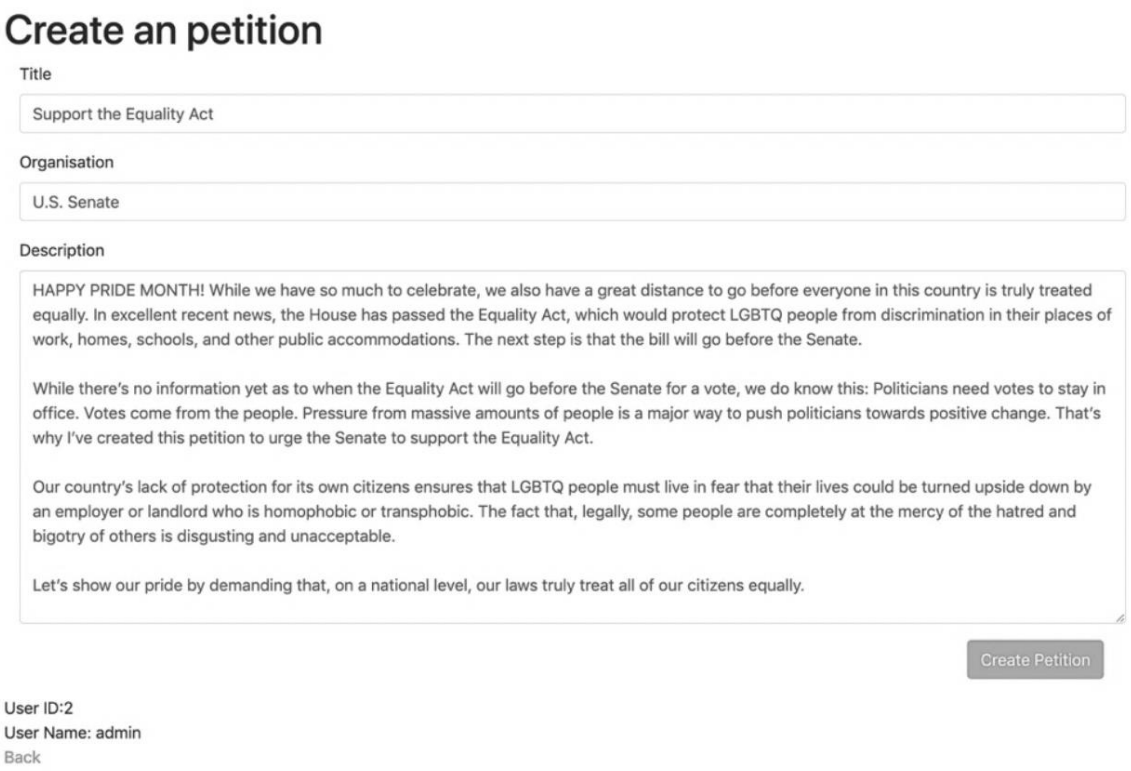

Рисунок 4.2.6 – Форма створення петиції

Якщо користувач заповнить форму не правильно система повідомить користувача, що саме він ввів не правильно.(Рисунок 4.2.7).

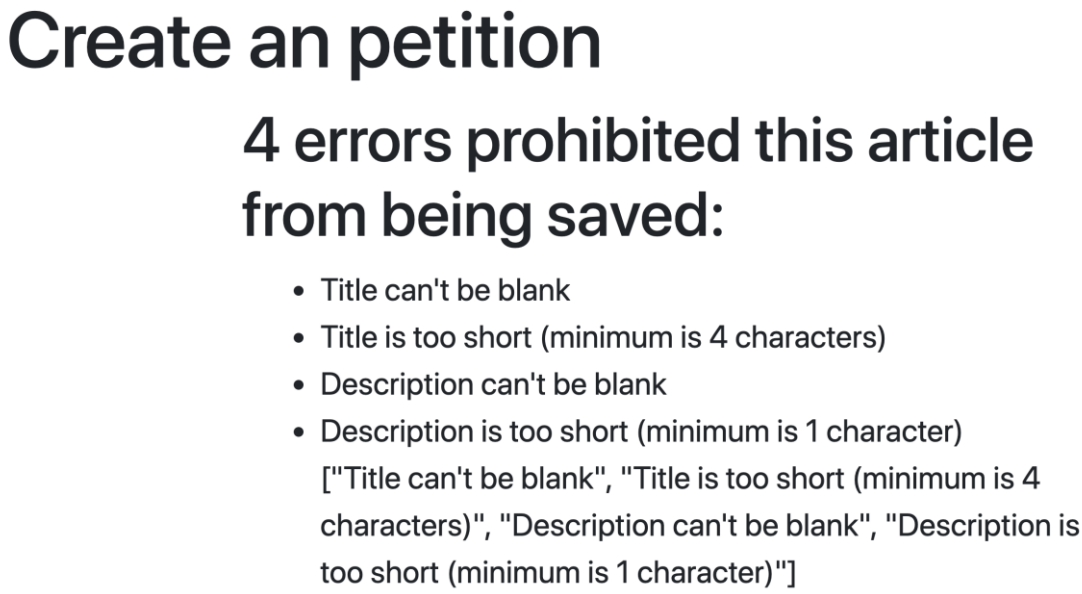

Рисунок 4.2.7 – Повідомлення створення петиції з помилками

4.2.5 Підтримання петиції

Користувач може не тільки створювати петиції, а також підтримувати петиції інших користувачів, для цього необхідно вибрати петицію зі списку та натиснути кнопку «Деталі»(Рисунок 4.2.8).

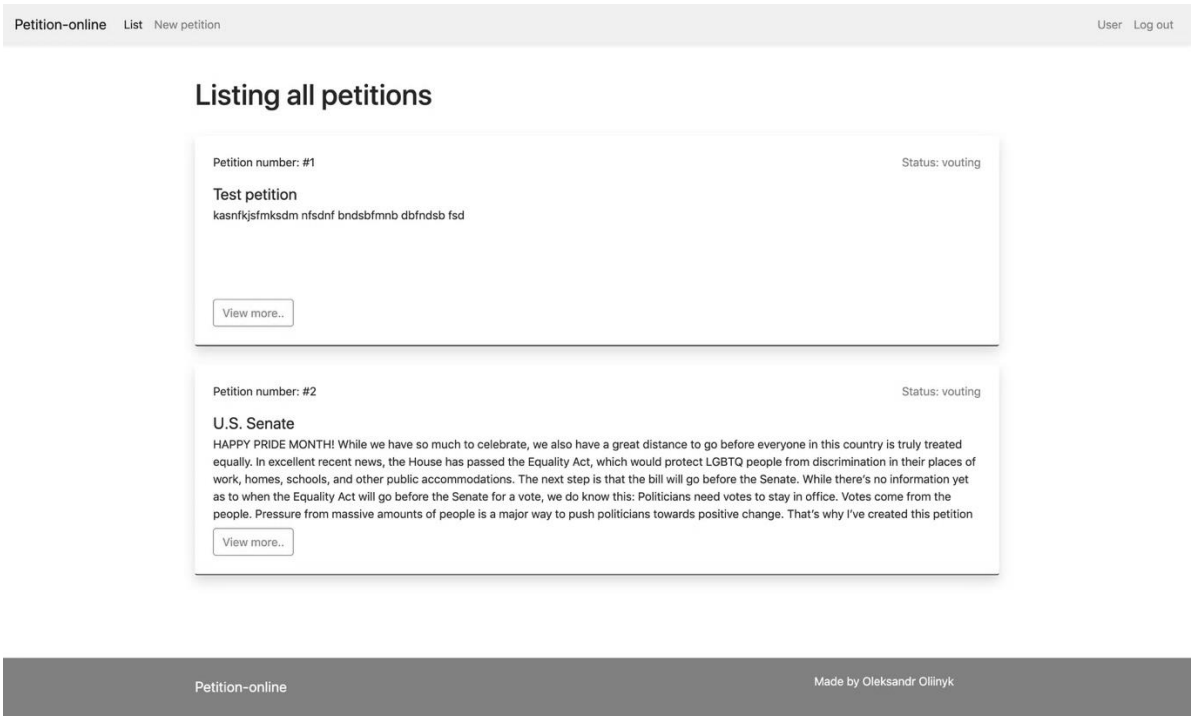

Рисунок 4.2.8 – Сторінка перегляду списку петицій

Після чого відкриється детальна інформація про петицію, ознайомившись з якою, користувач може її підтримати натиснувши кнопку «Підтримую» або якщо він змінив своє рішення кнопку «Не підтримую»(Рисунок 4.2.9).

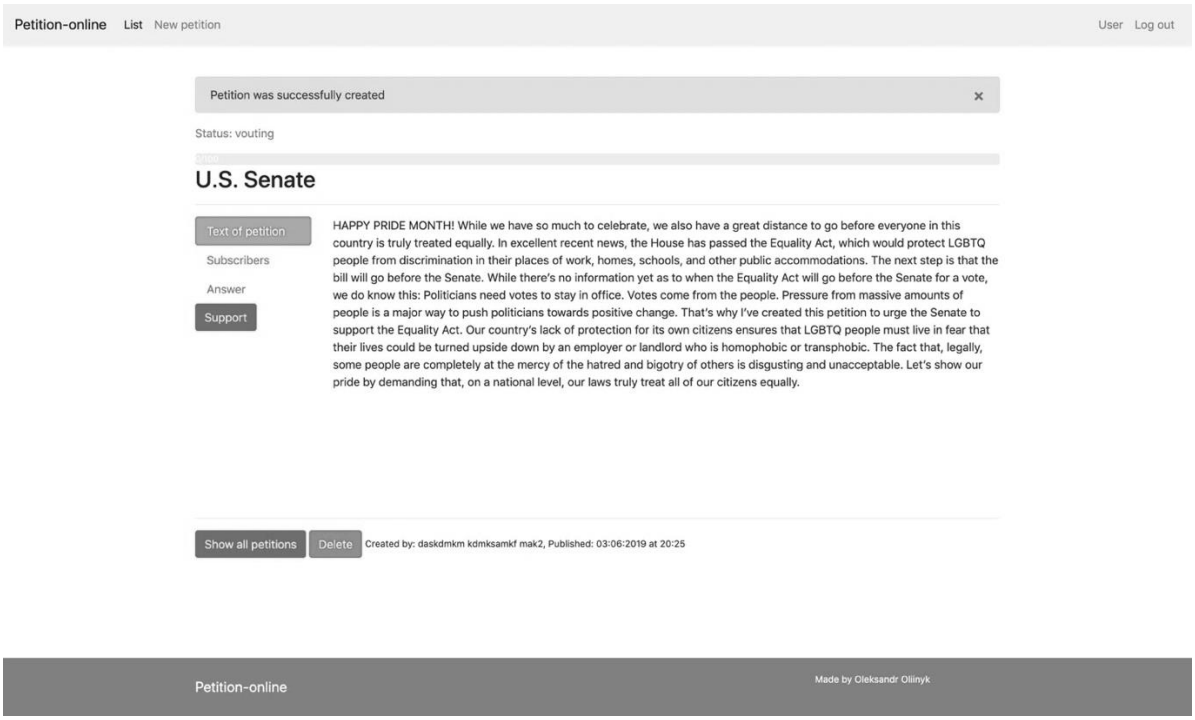

Рисунок 4.2.9 – Сторінка перегляду петиції

Також кожен користувач має можливість переглянути список користувачів, які підтримали дану петицію(Рисунок 4.2.10).

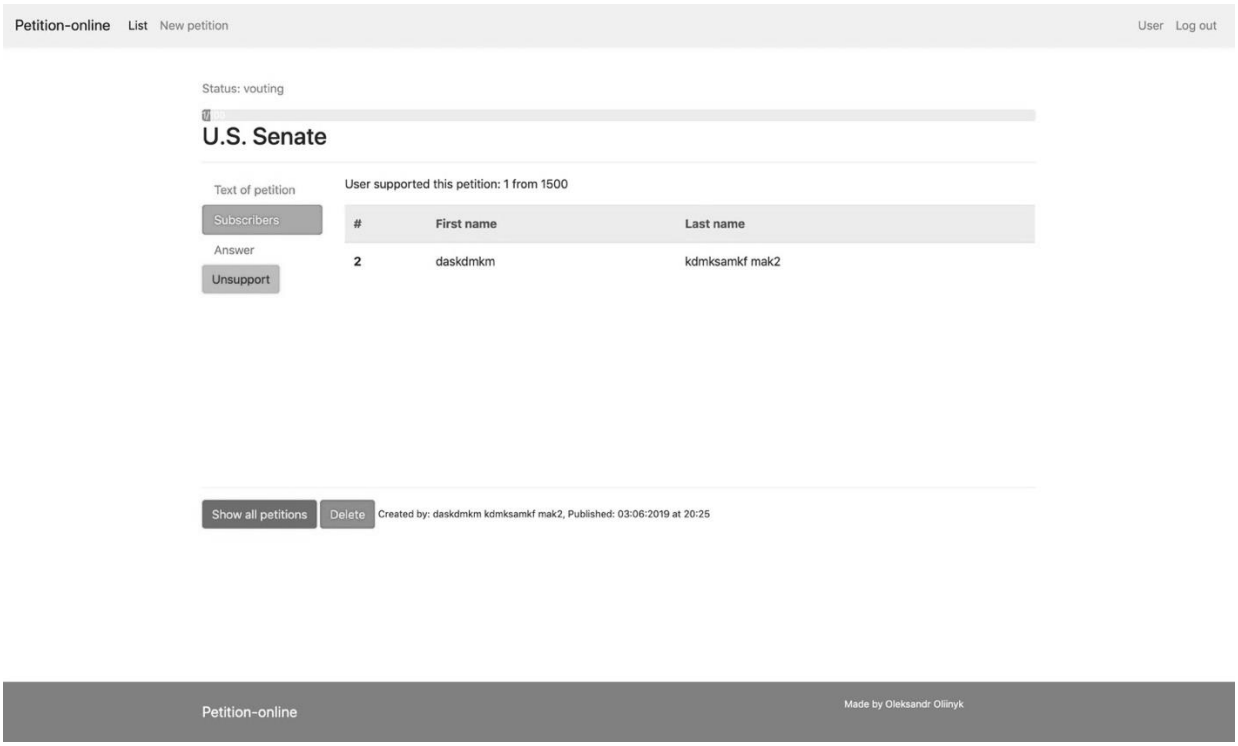

Рисунок 4.2.10 – Сторінка перегляду списку користувачів що підтримали дану петицію

4.2.6 Видалення петиції

Користувач має за собою право видалити свою петицію, якщо йому це необхідно, та він не хоче щоб вона була активною та приймала участь в голосуванні. Необхідно натиснути кнопку «Видалити» (Рисунок 4.2.11), після чого система попросить підтвердження у спливаюче вікні (Рисунок 4.2.12), після чого петиція буде видалена і користувач отримає повідомлення про це (Рисунок 4.2.13).

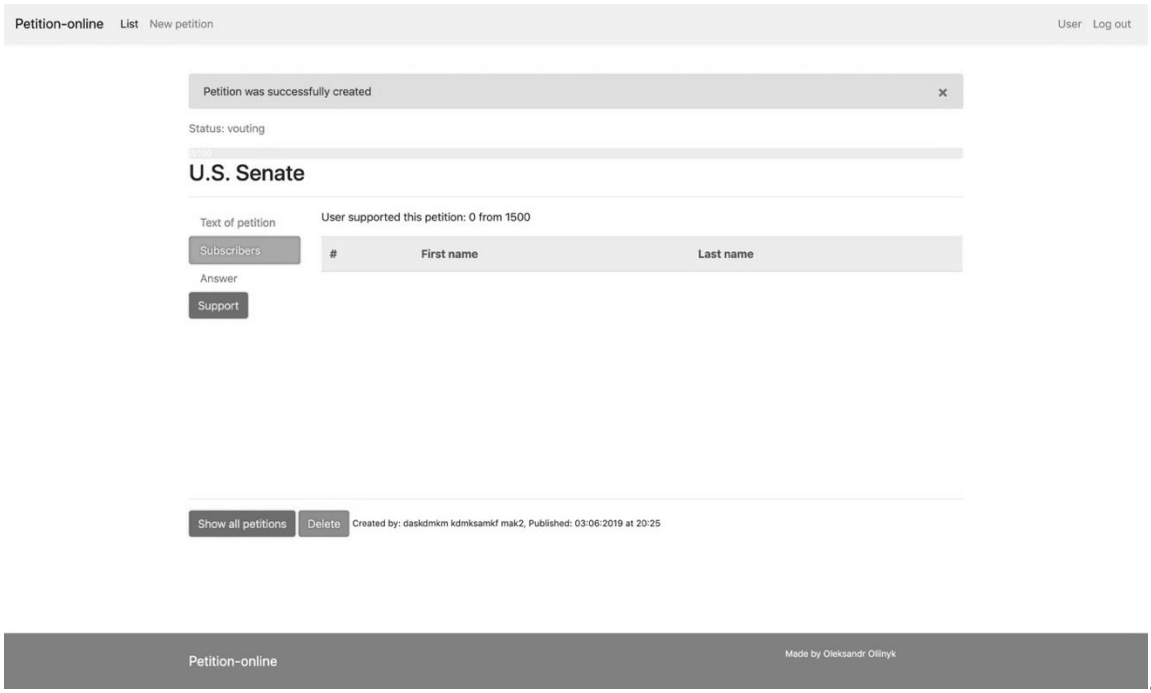

Рисунок 4.2.11 – Сторінка видалення петиції

Спливаюче вікно потрібно для того, щоб користувач не видалив петицію через необережність, а підтвердив свою дію в цьому вікні. Якщо користувач натисне «Відмінити», то нічого не відбудеться і петиція залишиться активною з тим статусом, який в неї був до цього.

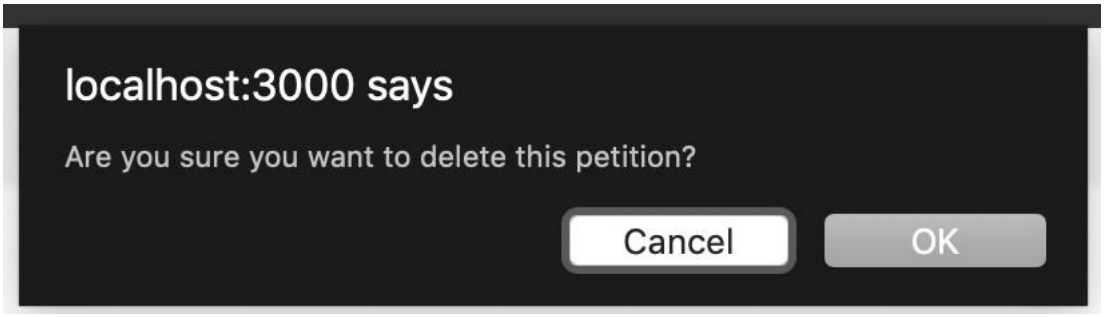

Рисунок 4.2.12 – Спливаюче вікно підтвердження видалення петиції

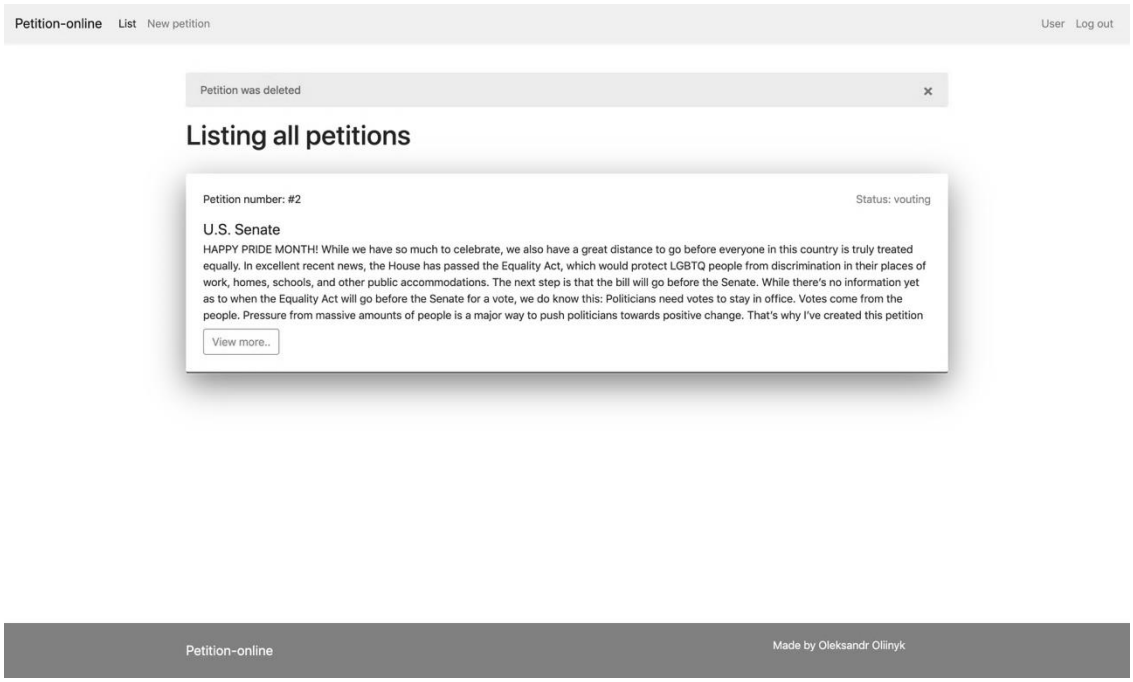

Рисунок 4.2.13 – Повідомлення про успішне видалення петиції

4.2.7 Відповідь на петицію

Коли петиція набирає необхідну кількість голосів, її статус змінюється та вона «відправляється» до користувача, який є представником компанії до, якої було направлено петицію. У користувача в списку петицій, які чекають відповідь з’являється дана петиція, натиснувши на яку, користувач потрапляє на сторінку цієї петиції. На сторінці петиції є форма для відповіді, після заповнення форми користувач повинен натиснути кнопку “Відправити” і відповідь на петицію буде розміщено на сторінці цієї петиції на вкладці «Відповідь» (Рисунок 4.2.14).

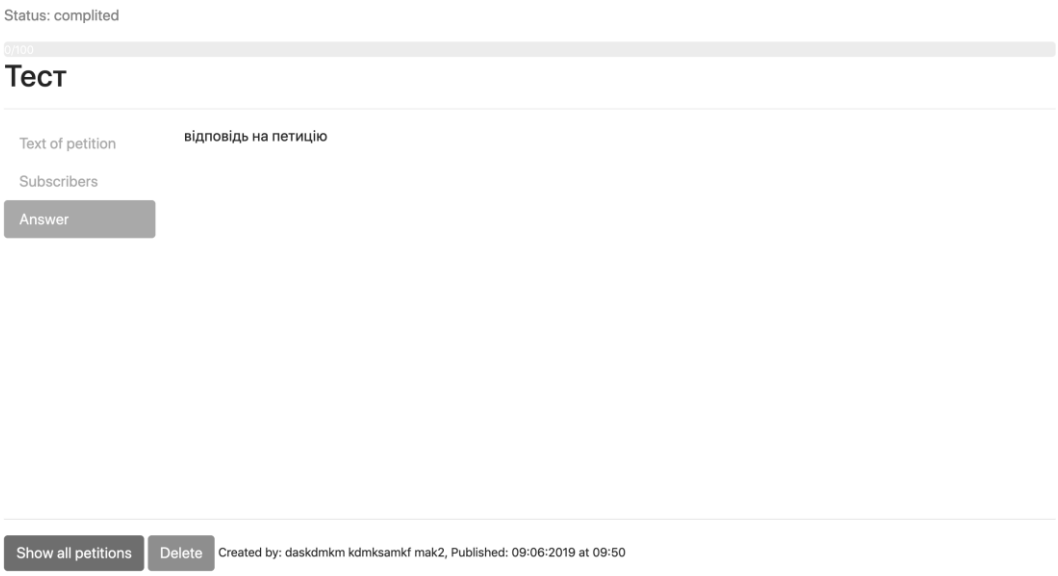

Рисунок 4.2.14 – Вкладка «Відповідь» на сторінці петиції

4.3 Висновок до розділу

В даному розділі було описано покрокове розгортання веб-застосунку на сервері зі списком команд, які необхідно виконати, що все запрацювало.

Було створено інструкцію користуванням веб-застосунком, на кожному етапі взаємодію користувача з системою, з детальним рисунками системи, що дає змогу наглядно показати користувачу, як взаємодіяти зі системою.

ВИСНОВКИ

В ході виконання даної дипломної роботи було проведено та проаналізовано предметну область розробки, а саме систему для збору та обробки електронних петицій. Було виділено подібні системи у обраній області та порівняно між собою. Для розробки системи було обрано найкращі частини готових систем, з метою створити покращену систему, яка не буде мати недоліків всіх існуючих рішень і також містити в собі їх переваги.

Було розроблено архітектуру системи. Обрано мову, яка найкраще підходить для реалізації поставлених задач, обрано паттерн на основі якого розроблено систему. Обрано мову для написання веб-інтерфейсу. Проаналізовано безпеку даних у системі.

Проаналізовано та обрано підходи та типи тестування системи, описано процес тестування, виконано тестування системи. Описано вимоги до середовища, а саме мінімальні системні вимоги.

Було розроблено та описано розгортання програмного продукту на сервері, з детальним пояснення та набором команд, які необхідно виконувати. Розроблено та описано детальну інструкцію користувача, на всіх етапах використання веб-застосунку.

ПЕРЕЛІК ПОСИЛАНЬ

1. What Is JavaScript and How Does It Work? [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.makeuseof.com/tag/what-is-javascript/.
2. Ruby on Rails Tutorial (Rails 5) [Електронний ресурс] – Режим доступу до ресурсу: https://www.railstutorial.org/book/beginning.
3. Software Testing Overview [Електронний ресурс] – Режим доступу до ресурсу:https://www.tutorialspoint.com/software_engineering/software_testing_overview.html.
4. РАЗВЕРТЫВАНИЕ RAILS-ПРИЛОЖЕНИЯ НА CAPISTRANO, NGINX И PUMA В UBUNTU 14.04 [Електронний ресурс] – Режим доступу до ресурсу: https://www.8host.com/blog/razvertyvanie-rails-prilozheniya-na-capistrano-nginx-i-puma-v-ubuntu-14-04/.
5. About MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/about/.
6. An Introduction to JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://javascript.info/intro.
7. About Ruby [Електронний ресурс] – Режим доступу до ресурсу: https://www.ruby-lang.org/en/about/.
8. WHAT IS A GEM? [Електронний ресурс] – Режим доступу до ресурсу: https://guides.rubygems.org/what-is-a-gem/.
9. Ruby on Rails 2.1 - Unit Testing [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/ruby-on-rails-2.1/rails-unit-testing.htm.
10. bcrypt [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Bcrypt.
11. How to Build a Blog with Ruby on Rails [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@deallen7/ruby-on-rails-app-build-blog-3d9975a999ae.

12. BankID [Електронний ресурс] – Режим доступу до ресурсу: https://bankid.org.ua.

13. Cascading Style Sheets [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Cascading_Style_Sheets.

14. Integration Testing: What is, Types, Top Down & Bottom Up Example [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/integration-testing.html.

15. Jay Sridhar. What Is JavaScript and How Does It Work? [Електронний ресурс] / Jay Sridhar. – 2017. – Режим доступу до ресурсу: https://www.makeuseof.com/tag/what-is-javascript/.

ДОДАТОК А. СХЕМА СТРУКТУРНА ВАРІАНТІВ ВИКОРИСТАНЬ

ДОДАТОК Б. СХЕМА СТРУКТУРНА СТАНІВ СИСТЕМИ

ДОДАТОК В. СХЕМА БАЗИ ДАНИХ